PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-185548
(43)Date of publication of application : 15.07.1997

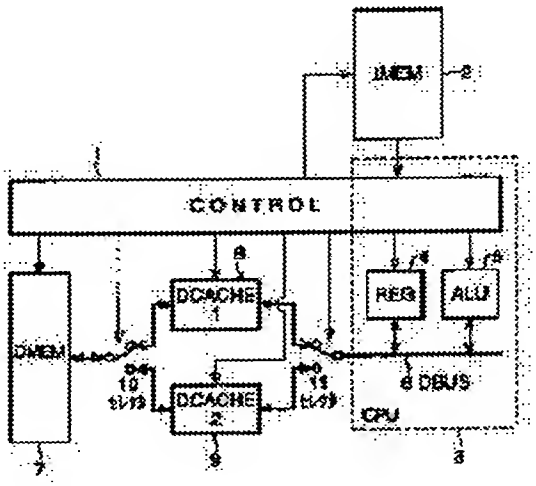
(51)Int.Cl. G06F 12/08
G06F 12/08
G06F 12/12
H03M 7/40

(21)Application number : 07-343170
(22)Date of filing : 28.12.1995

(71)Applicant : SONY CORP
(72)Inventor : OKI MITSUHARU
TOTSUKA TAKUSHI

(54) CACHE MEMORY CONTROL METHOD, VARIABLE LENGTH ENCODING METHOD AND VARIABLE LENGTH DECODING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To execute efficient encoding or decoding corresponding to code length.
SOLUTION: When prescribed input data is encoded, data to be referred to is stored in a data memory 7 and data corresponding to the code with comparatively short code length, which requires high speed reference, among data stored in the data memory 7 is transferred to a data cache 8 and is locked. ALU 5 constituting CPU 3 reads data stored in the data cache 8 through a data bus 6, a register file 4 and a selector 11 without a cache error, and encodes the input data based on the data.



LEGAL STATUS

[Date of request for examination] 13.09.2002
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

引用例１の写し

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平9－185548
(43)公開日　平成９年(1997)７月15日

(51)Int.Cl.[6]		識別記号	庁内整理番号	Ｆ１			技術表示箇所
Ｇ０６Ｆ	12/08	３１０	7623－5B	Ｇ０６Ｆ	12/08	３１０Ｚ	
			7623－5B			Ｂ	
	12/12		7623－5B		12/12	Ｄ	
Ｈ０３Ｍ	7/40		9382－5K	Ｈ０３Ｍ	7/40		

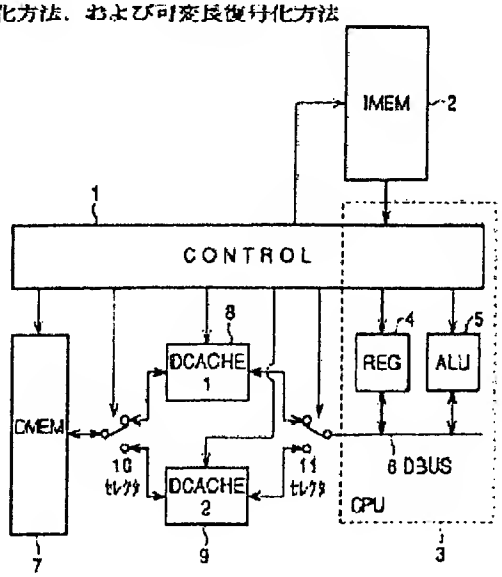
審査請求　未請求　請求項の数６　ＯＬ　（全 18 頁）

(21)出願番号　特願平7－343170
(22)出願日　平成７年(1995)12月28日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　大木　光晴
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(72)発明者　戸塚　卓志
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　弁理士　船本　義雄

(54)【発明の名称】　キャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法

(57)【要約】
【課題】　符号長に応じた効率的な符号化または復号化を可能にする。
【解決手段】　所定の入力データを符号化するとき参照するデータをデータメモリ７に記憶させ、データメモリ７に記憶されたデータのうち、高速な参照が必要な比較的短い符号長の符号に対応するデータをデータキャッシュ８に転送し、ロックする。ＣＰＵ３を構成するＡＬＵ５は、データ用バス６、レジスタファイル４、およびセレクタ１１を介して、データキャッシュ８に記憶されたデータをキャッシュミスなしで高速に読み出し、それに基づいて、入力データを符号化する。



【特許請求の範囲】
【請求項 1】　データを第１のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラム を第２のメモリに記憶し、前記データを参照して、前記プログラム を実行する場合において、前記キャッシュメモリを制御するキャッシュメモリ制御方法において、
前記データのうち、所定の時間内で参照すべきものを予め前記キャッシュメモリに転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記データの所定のものを前記キャッシュメモリに常駐させることを特徴とするキャッシュメモリ制御方法。
【請求項 2】　データを第１のメモリに記憶し、所定のプログラム を第２のメモリに記憶し、前記プログラム をキャッシュメモリに転送し、前記データを参照して、前記プログラム を実行する場合において、前記キャッシュメモリを制御するキャッシュメモリ制御方法において、
前記プログラム のうち、所定の時間内で実行すべきものを予め前記キャッシュメモリに転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラム の所定のものを前記キャッシュメモリに常駐させることを特徴とするキャッシュメモリ制御方法。
【請求項 3】　データを第１のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラム を第２のメモリに記憶し、前記データを参照して、前記プログラム を実行し、所定の入力データに対して可変長符号化を行い、符号化された圧縮データを生成する可変長符号化方法において、
前記第１のメモリに、前記入力データに対応する符号化された前記圧縮データを記憶させ、
前記第１のメモリに記憶された前記圧縮データのうち、前記圧縮データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記圧縮データを前記キャッシュメモリに常駐させることを特徴とする可変長符号化方法。
【請求項 4】　データを第１のメモリに記憶し、所定のプログラム を第２のメモリに記憶し、前記プログラム をキャッシュメモリに転送し、前記データを参照して、前記プログラム を実行し、所定の入力データに対して可変長符号化を行い、符号化された圧縮データを生成する可変長符号化方法において、
前記入力データに対応する前記圧縮データを生成するプログラム を前記第２のメモリに記憶させ、
前記第２のメモリに記憶された前記プログラム のうち、前記プログラム が生成する前記圧縮データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラム を前記キャッシュメモリに常駐させることを特徴とする可変長符号化方法。
【請求項 5】　データを第１のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラム を第２のメモリに記憶し、前記データを参照して、前記プログラム を実行し、所定の圧縮された入力データに対して可変長復号化を行い、前記入力データが復号化された復号化データを生成する可変長復号化方法において、
前記第１のメモリに、圧縮された前記入力データに対応する前記復号化データを記憶させ、
前記第１のメモリに記憶された前記復号化データのうち、対応する前記入力データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記復号化データを前記キャッシュメモリに常駐させることを特徴とする可変長復号化方法。
【請求項 6】　データを第１のメモリに記憶し、所定のプログラム を第２のメモリに記憶し、前記プログラム をキャッシュメモリに転送し、前記データを参照して、前記プログラム を実行し、所定の圧縮された入力データに対して可変長復号化を行い、前記入力データが復号化された復号化データを生成する可変長復号化方法において、
前記入力データに対応する前記復号化データを生成するプログラム を前記第２のメモリに記憶させ、
前記第２のメモリに記憶された前記プログラム のうち、前記プログラム が生成する前記復号化データに対応する前記入力データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、
前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラム を前記キャッシュメモリに常駐させることを特徴とする可変長復号化方法。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、キャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法に関し、例えば、画像データを符号化したり、符号化された画像データを復号化する場合に用いて好適なキャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法に関する。
【０００２】
【従来の技術】現在、画像圧縮の方式として、一番よく使われている方式が、ＭＰＥＧ２（Moving Picture Experts Group phase 2）である。ＭＰＥＧ２は、ＩＳＯ（International Organization for Standardization:

国際標準 化機構）が提案した画像圧縮方式であ り、詳細は、文献ＩＳＯ／ＩＥＣ１３８１８－２に述べられている。

【０００３】ＭＰＥＧ２の符号化方法においては、入力画像データに対して２次元８×８ＤＣＴ（離散コサイン変換）をした後、量子化を行い、量子化されたデータをジグザグスキャンデータ列に並べ直し、ＶＬＣ（バリアブルレングスコーディング）をして、圧縮を行っている。

【０００４】量子化後のデータは２次元８×８行列上のデータとなるが、ジグザグスキャンデータ列とは、図５に示す矢印の順番にデータをスキャンして１列に並べることであ る。符号化方法のＶＬＣでは、ジグザグスキャンデータ列をｒｕｎとｌｅｖｅｌの２つより成るデータ（ｒｕｎ，ｌｅｖｅｌ）に変換し、この（ｒｕｎ，ｌｅｖｅｌ）のデータを符号化して圧縮している。ここで、ｒｕｎとは、ノンゼロ（有効データ）までのデータの個数、即ち、０（無効データ）がいくつ連続しているかをカウントした値であ り、ｌｅｖｅｌとは、ｒｕｎだけ０が続いた後のノンゼロの値であ る。

【０００５】図５の場合、ジグザグスキャンデータ列は、”０，０，０，１，０，１８，０，－２，０，０，．．．，０”となるので、０が３個の後、１、その後、０が１個、その後、１８、その後、０が１個、その後、－２、そして残りは０となる。従って、図６に示すように、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）、（１，１８）、および（１，－２）となる。ＶＬＣでは、上記３個の（ｒｕｎ，ｌｅｖｅｌ）を順番に「（ｒｕｎ，ｌｅｖｅｌ）対（符号）」のテーブルを参照しながら、符号化していく。

【０００６】図７乃至図１０に、上記「（ｒｕｎ，ｌｅｖｅｌ）対（符号）」のテーブルを載せておく。このテーブルから分かるように、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）は”００１１１０”と符号化される。テーブル上で（３，１）に対応する符号は”００１１１ｓ”と書かれているが、ｓとはｌｅｖｅｌが正のときは０、負のときは１であ る。つまり、ｓとはｌｅｖｅｌの符号であ る。

【０００７】また、（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）は、”０００００００００００１００００”と符号化され、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）は、”０００１１０１”と符号化される。従って、量子化後のデータが図５に示すような場合、図１１に示すような符号”００１１１００００００００００１００００００１１０１”となる。

【０００８】また、図７乃至図１０に載っていない（ｒｕｎ，ｌｅｖｅｌ）に対しては、Ｅｓｃａｐｅ符号の後に、図１２に示す符号を割り当てる。Ｅｓｃａｐｅ符号は図７に示すように、”０００００１”という符号である。この後、ｒｕｎに対応する符号を図１２から選出して、この符号をＥｓｃａｐｅ符号の後に付け加える。図１２に示す符号は、ｒｕｎを６ビット２進数表示したときのビットそのものであ る。この後、ｌｅｖｅｌに対応する符号を図１２から選出して、この符号をｒｕｎの符号の後に付け加える。図１２に示す符号は、ｌｅｖｅｌを１２ビット２の補数表示したときのビットそのものであ る。ちなみに、図７乃至図１０に載っていない（ｒｕｎ，ｌｅｖｅｌ）に対しては、Ｅｓｃａｐｅ符号も含めると２４ビットの長さの符号が割り当てられる。

【０００９】また、ＭＰＥＧ２において、（ｒｕｎ，ｌｅｖｅｌ）の一番最初がｒｕｎ＝０のときは特殊な符号が割り当てられており、さらに、（ｒｕｎ，ｌｅｖｅｌ）がこれ以上ないことを示すＥｎｄＯｆＢｌｏｃｋという符号もあ るが、本発明には関係がないので、ここではその説明は省略する。

【００１０】以下、図１３乃至図１５を参照して、従来の情報処理装置の構成、およびこの装置により実行されるバリアブルレングスコーディング（ＶＬＣ）について詳細に説明する。

【００１１】図１３は、従来の情報処理装置の一例の構成を示すブロック図であ る。図１３に示した装置は、一般的な情報処理装置であ る。

【００１２】図１３において、制御回路（ＣＯＮＴＲＯＬ）１は、装置全体を制御するようになされている。インストラクションメモリ（ＩＭＥＭ）２は、所定のインストラクション（プログラム のコード等）を記憶する。データメモリ（ＤＭＥＭ）７は、図７乃至図１０、および図１２に示したテーブルのデータ等を記憶する。レジスタファイル（ＲＥＧ）４は、制御回路１からの指令に基づいて、後述するセレクタ１０、データキャッシュ（ＤＣＡＣＨＥ）１またはデータキャッシュ２、セレクタ１１、およびデータ用バス（ＤＢＵＳ）６を介してデータメモリ７より供給されたデータを記憶し、算術演算回路（ＡＬＵ）５に供給するようになされている。データ用バス６は、レジスタファイル４と算術演算回路５との間で、データの受渡しを行うようになされている。

【００１３】レジスタファイル４、算術演算回路５、データ用バス６、および制御回路１の一部分（点線で囲まれた部分）は、いわゆるＣＰＵ（Central Processing Unit）３を構成している。

【００１４】セレクタ１０は、制御回路１によって制御され、データメモリ７からのデータをデータキャッシュ８またはデータキャッシュ９のいずれかに供給するようになされている。データキャッシュ８およびデータキャッシュ９は高速にデータの読み書きが可能なメモリであり、データキャッシュ８は、セレクタ１０を介してデータメモリ７からデータが供給されたとき、制御回路１の制御により、データメモリ７からのデータを記憶する。また、データキャッシュ９は、セレクタ１０を介してデータメモリ７からデータが供給されたとき、制御回路１

の制御により、そのデータを記憶するようになされている。

【００１５】セレクタ１１は、制御回路１の制御に基づいて、データキャッシュ８またはデータキャッシュ９のいずれかに記憶されたデータをデータ用バス６を介して、レジスタ４に供給するようになされている。

【００１６】インストラクションメモリ２には、処理すべきプログラム の内容（コード）が格納されている。制御回路１からの制御により、インストラクションメモリ２内のプログラム のコードが順番に読み出され、読み出されたコードを制御回路１が解釈して、適切なデータをデータメモリ７から読み出し、レジスタファイル４と算術演算回路５を使用して所定の演算を行い、データメモリ７に演算結果を戻すようになされている。

【００１７】上記処理すべきプログラム の内容とは、図１４のフローチャートを参照して後述するように、例えば、ステップＳ３，Ｓ４、およびＳ５に対応するプログラム であり、データメモリ７上にあるデータ（ｒｕｎ，ｌｅｖｅｌ）をレジスタファイル４にロードしたり、ｒｕｎの値が３１を越えているか否か、あるいは、ｌｅｖｅｌの絶対値が４０を越えているか否かを算術演算回路５に判定させたり、あるいは、算術演算回路５に（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎを計算させ、その計算結果を変数ｐｔｒの値とし、それをデータメモリ７上において変数ｐｔｒというデータが格納されている番地にストアさせたりすることである。

【００１８】また、図１４のステップＳ６に対応するプログラム として、データメモリ７上にある、ａ［０］乃至ａ［３２×４０－１］、およびｂ［０］乃至ｂ［３２×４０－１］の内、ａ［ｐｔｒ］、およびｂ［ｐｔｒ］のデータをレジスタファイル４にロードすることである。さらにまた、ステップＳ７に対応するプログラム として、上記レジスタファイル４にロードされたデータｂ［ｐｔｒ］の値が、２４であるか否かを算術演算回路５に判定させることである。

【００１９】次に、図１４のフローチャートを参照して、図１３に示した情報処理装置の動作について説明する。

【００２０】最初、ステップＳ１において、初期設定が行われる。即ち、図１５に示したフローチャートで表される初期設定のサブルーチンが実行される。図１５のステップＳ２１においては、図７乃至図１０に示したテーブルのデータ（符号）を格納しておく領域ａ［０］乃至ａ［３２×４０－１］と、その符号がどれほどの長さ（ビット長）であるかを記憶しておくための領域ｂ［０］乃至ｂ［３２×４０－１］がデータメモリ７上に確保される。

【００２１】そして、図７乃至図１０に示す（ｒｕｎ，ｌｅｖｅｌの絶対値）に対応する各符号が、ｅ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］に上位側につめてそれぞれ格納され、それらの符号の長さ（ビット長）が、ｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］にそれぞれ格納される。対応する（ｒｕｎ，ｌｅｖｅｌの絶対値）が図７乃至図１０に示したテーブルにないために、またデータ（符号）が格納されていない残りのｂ［ｊ］（ここで、ｊは、まだ符号が格納されていない領域を示す）には、符号の最大のビット長である値２４が格納される。

【００２２】但し、図７乃至図１０に示す符号の最後のｓについては無視される。つまり、例えば、（ｒｕｎ，ｌｅｖｅｌの絶対値）＝（１，２）の符号は”０００１１０ｓ”であるから、ｓは無視され、ａ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］＝ａ［３３］＝”０００１１０”（バイナリデータ、上位ビット側につめられる）とされ、ｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］＝ｂ［３３］＝６とされる。

【００２３】また、図７乃至図１０には、ｒｕｎが３２以上のものはなく、また、ｌｅｖｅｌの絶対値が４１以上のものもないので、（ｒｕｎ，ｌｅｖｅｌの絶対値）が決まれば、（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎは一意に決まる。また、異なる２つの（ｒｕｎ，ｌｅｖｅｌの絶対値）においては、（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎの値が同一となることはない。さらに、図７乃至図１０に示されている符号は全て２３ビット以下であり、ｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］は２３以下である。

【００２４】ステップＳ２１における初期化処理が終了すると、リターンし、図１４のステップＳ２に進む。

【００２５】ステップＳ２においては、制御回路１により、読み込むべき（ｒｕｎ，ｌｅｖｅｌ）があるか否かが判定される。読み込むべき（ｒｕｎ，ｌｅｖｅｌ）があると判定された場合、ステップＳ３に進み、（ｒｕｎ，ｌｅｖｅｌ）が読み込まれる。

【００２６】次に、ステップＳ４において、制御回路１に制御された算術演算回路５により、読み込まれたｒｕｎの値が３１を越えているか、または読み込まれたｌｅｖｅｌの絶対値が４０を越えているか否かが判定される。ｒｕｎの値が３１を越えているか、またはｌｅｖｅｌの絶対値が４０を越えている場合、ステップＳ９に進む。図７乃至図１０に示したテーブルには、ｒｕｎが３１を越えるものはなく、また、ｌｅｖｅｌの絶対値が４０を越えるものもない。従って、このステップＳ４において、ｒｕｎの値が３１を越えているか、またはｌｅｖｅｌの値が４０を越えていると判定された場合、ステップＳ９において、図１２に示した符号が割り当てられることになる。

【００２７】一方、ｒｕｎの値が３１を越えておらず、かつｌｅｖｅｌの絶対値が４０を越えていない場合、ステップＳ５に進み、変数ｐｔｒに（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎが代入される。そして、ステップ

ＳＳにおいて、変数ｃｏｄｅｔｍｐにｓ［ｐｔｒ］が、また、変数ｃｏｄｅｌｅｎｇｔｈにｂ［ｐｔｒ］が代入される。

【００２８】次に、ステップＳ７に進み、制御回路１に制御された算術演算回路５により、ｃｏｄｅｌｅｎｇｔｈの値が２４であるか否かが判定される。ｃｏｄｅｌｅｎｇｔｈの値が２４であると判定された場合、ステップＳ９に進む。

【００２９】ｒｕｎの値が３１以下であり、かつ、ｌｅｖｅｌの絶対値が４０以下であっても、図７乃至図１０に示したテーブルに載っていないものがある。ステップＳ１のサブルーチン（図１５）内のステップＳ２１における初期設定処理において、図７乃至図１０に示したテーブルに載っている（ｒｕｎ，ｌｅｖｅｌ）に対応するｂ［ｋ］（ここで、ｋ＝（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ）は全て２３以下であり、テーブルに載っていない（ｒｕｎ，ｌｅｖｅｌ）に対応するｂ［ｊ］（ここで、ｊはｋ以外で可能な整数）は全て２４であるように設定されている。従って、この判定処理で、ｂ［ｊ］が２４であると判定された場合は、図１２に示す符号を割り当てる必要がある。

【００３０】ステップＳ９においては、まず、Ｅｓｃａｐｅ符号である”０００００１”が出力され、次に、図１２に示したように、ｒｕｎの値を６ビット２進数で表現したビット列が出力され、さらに、ｌｅｖｅｌの値を１２ビット２の補数で表現したビット列が出力される。その後、ステップＳ２に戻り、ステップＳ２以降の処理が繰り返し実行される。

【００３１】一方、ｃｏｄｅｌｅｎｇｔｈの値が２４でないと判定された場合、すなわち、ステップＳ３において読み込まれた（ｒｕｎ，ｌｅｖｅｌ）に対応する符号が、図７乃至図１０に示したテーブルに載っている場合のみ、ステップＳ８の処理が実行される。ステップＳ８においては、ｃｏｄｅｔｍｐの上位ビット側からｃｏｄｅｌｅｎｇｔｈビット分だけ出力される。ｓ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］の上位ビット側からｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］の長さ分には、図７乃至図１０のテーブルに示した符号が格納されているので、これに基づいて、ステップＳ３で読み込まれた（ｒｕｎ，ｌｅｖｅｌ）に対応する符号が出力される。

【００３２】但し、図７乃至図１０のテーブルに示された符号の最後のｓに対応する信号がないので、ｌｅｖｅｌの値が正であれば０を、負であれば１を最後に出力する。例えば、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）がステップＳ３で読み込まれたときは、ｓ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］＝ｓ［３３］＝０００１１０（バイナリデータ）とｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］＝ｂ［３３］＝６が、ステップＳ６においてｃｏｄｅｔｍｐとｃｏｄｅｌｅｎｇｔｈにそれぞれ代入されるので、ステップＳ８において、ｃｏｄｅｔｍｐの上位６ビットである”０００１１０”が読み出され、そして、その後に、ｌｅｖｅｌの符号である１が付け加えられて、出力される。

【００３３】ステップＳ８の処理が終了した後、ステップＳ２に戻り、次の（ｒｕｎ，ｌｅｖｅｌ）の処理に移る。

【００３４】図５、および図６に示した例では、最初にステップＳ３において（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）が読み込まれ、ステップＳ４乃至Ｓ８の処理が行われ、次にステップＳ３において（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）が読み込まれ、ステップＳ４乃至Ｓ８の処理が行われ、さらにステップＳ３において（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）が読み込まれ、ステップＳ４乃至Ｓ８の処理が行われる。そして、図１１に示す符号が出力される。

【００３５】このようにして、ＶＬＣが実行される。

【００３６】図１３に示した情報処理装置には、上述したようにデータキャッシュ（ＤＣＡＣＨＥ）８、およびデータキャッシュ（ＤＣＡＣＨＥ）９があり、データメモリ７が直接、データ用バス６につながっていない。つまり、データメモリ７は、データキャッシュ８あるいはデータキャッシュ９を介してデータ用バス６につながっている。

【００３７】データメモリ７は大容量のメモリであり、アクセスするのに時間がかかり過ぎる。そこで、データメモリ７の一部分をデータキャッシュ８あるいはデータキャッシュ９にコピーする。データキャッシュ８とデータキャッシュ９は小容量のメモリであり、アクセス時間は短い。従って、使用するデータがデータキャッシュ８あるいはデータキャッシュ９にコピーされていれば、このデータをデータキャッシュ８あるいはデータキャッシュ９から高速に読み出し、データ用バス６を介してレジスタファイル４にロードすることができる。

【００３８】このように、データキャッシュ８またはデータキャッシュ９を使うことにより、データを高速に読み出して、レジスタファイル４にロードすることができる。但し、データキャッシュ８およびデータキャッシュ９は小容量であり、データメモリ７の全てのデータをコピーすることは出来ないので、もし、ロードする際に、データキャッシュ８あるいはデータキャッシュ９にデータのコピーが見つからなければ、これをキャッシュミスと呼び、データメモリ７からデータを読み出すと共に、データキャッシュ８あるいはデータキャッシュ９にそのコピーを置く操作を行なう。この間、図１３に示した装置は計算を一時中断する。そして、コピーし終わった時点で計算を再開する。

【００３９】セレクタ１０は、データキャッシュ８およびデータキャッシュ９のどちらをデータメモリ７にアクセス（接続）させるかをセレクトするようになされてい

る。セレクタ１１は、データキャッシュ８およびデータキャッシュ９のどちらをデータ用バス６にアクセス（接続）させるかをセレクトするようになされている。

【００４０】データキャッシュは、最近では１つではなく２つ（この実施例の場合、データキャッシュ８およびデータキャッシュ９）に別れている。即ち、２－ｗａｙ－ｓｅｔ－ａｓｓｏｃｉａｔｉｖｅの構成をとっている。データメモリ７におけるある特定の番地のデータは、データキャッシュ８の特定の番地にしかコピーされない。同様に、データメモリ７におけるある特定の番地のデータは、データキャッシュ９の特定の番地にしかコピーされない。つまり、データメモリ７におけるある特定の番地（ＡＤＲＳ）のデータは、データキャッシュ８もしくはデータキャッシュ９の特定の番地（ＣＡＤＲＳ）にしかコピーされない。

【００４１】通常、ＣＡＤＲＳ＝（ＡＤＲＳ）ｍｏｄ（Ｃｓ）なる式を用いて、データキャッシュ上の番地（ＣＡＤＲＳ）を算出する。ここでＣｓはキャッシュ（データキャッシュ８とデータキャッシュ９）の総バイト数を２で割ったものである。したがって、データメモリ７のある特定の番地（ＡＤＲＳ）に対応するデータのコピーを、データキャッシュ８またはデータキャッシュ９から読み出すには、データキャッシュ８もしくはデータキャッシュ９の所定の番地（ＣＡＤＲＳ）の箇所（キャッシュエントリと呼ぶ）に、データメモリ７の番地（ＡＤＲＳ）に対応するデータのコピーが存在するか否かをチェックすればよい。そして、データキャッシュ８またはデータキャッシュ９のいずれにもこのデータのコピーが存在しなければキャッシュミスとなる。

【００４２】このように２－ｗａｙ－ｓｅｔ－ａｓｓｏｃｉａｔｉｖｅでは２箇所のチェックで済み、ｆｕｌｌｙ－ａｓｓｏｃｉａｔｉｖｅのようにデータキャッシュの全ての番地をチェックする必要はない。これにより、このチェックをすばやく行うことが出来る。

【００４３】２－ｗａｙ－ｓｅｔ－ａｓｓｏｃｉａｔｉｖｅの構成では、データメモリ７内の特定の３つ以上のデータが、データキャッシュ８の１つの共通の番地にしかコピーされないことがありうる。同様に、データメモリ７内の特定の３つ以上のデータが、データキャッシュ９の１つの共通の番地にしかコピーされないことがありうる。つまり、データメモリ７上の３個以上の異なる番地（Ａ１、Ａ２、Ａ３、．．．）に対して、同一の番地（ＣＡＤＲＳ）をもつキャッシュエントリがデータキャッシュ８およびデータキャッシュ９に割り当てられることがある。

【００４４】一般に、Ａｎ＝ＡＤＲＳ＋ｎ×Ｃｓ（ｎは整数）で与えられる番地（Ａｎ）に対して、番地（ＡＤＲＳ）と同一のキャッシュエントリが割り当てられる。ところが、同一の番地（ＣＡＤＲＳ）をもつキャッシュエントリはデータキャッシュ８に１個、データキャッシュ９に１個の合計２個しかないので、３つのデータを順次、ロードしたいときには、１つ目のデータはデータキャッシュ８にコピーされていて、２つ目のデータはデータキャッシュ９にコピーされていて、キャッシュミスなしに１つ目と２つ目のデータをロードできる可能性があるが、３つ目のデータは必ずキャッシュミスを起こす。なぜなら、３つ目のデータがコピーされるべき番地は、既に１つ目と２つ目のデータによってそれぞれ占有されており、３つ目のデータはデータキャッシュ８とデータキャッシュ９には絶対にコピーされないからである。

【００４５】このようなときには、３つ目のデータをデータキャッシュ８の１つ目のデータの上、あるいはデータキャッシュ９の２つ目のデータの上にオーバーライトすることによりコピーする。そして、データキャッシュ８あるいはデータキャッシュ９にコピーされた３つ目のデータをロードする。このように、コピーしようとする番地が既に他のデータにより占有されている場合、キャッシュミスを起こすが、このとき、データキャッシュ８またはデータキャッシュ９のいずれにオーバーライトするかは、制御回路１において、自動的に決定される。このとき、制御回路１は、Ｒａｎｄｏｍ方式やＬＲＵ（ｌｅａｓｔ－ｒｅｃａｎｔｌｙ　ｕｓｅｄ）方式等の決定方法に従って、この決定を行うことができる。

【００４６】キャッシュ、２－ｗａｙ－ｓｅｔ－ａｓｓｏｃｉａｔｉｖｅ、ｆｕｌｌｙ－ａｓｓｏｃｉａｔｉｖｅ、Ｒａｎｄｏｍ方式、およびＬＲＵ方式については、一般的に知られているので、以上で説明を終わり、その詳細な説明は省略する。詳細については、例えば、Ｍｏｒｇａｎ　Ｋａｕｆｍａｎｎ　Ｐｕｂｌｉｓｈｅｒｓ，Ｉｎｃ．より発刊されている「ＣＯＭＰＵＴＥＲ　ＡＲＣＨＩＴＥＣＴＵＲＥ　Ａ　ＱＵＡＮＴＩＴＡＴＩＶＥ　ＡＰＰＲＯＡＣＨ（著者：ＪＯＨＮ　ＬＨＮＮＥＳＳＹ，ＤＡＶＩＤ　Ａ　ＰＡＴＴＥＲＳＯＮ）」などを参照のこと。

【００４７】また、最近では、キャッシュミスを起こした場合、データキャッシュ８あるいはデータキャッシュ９のどちらにコピーするかを、インストラクションメモリ２のプログラムの中に明示しておくことが出来るようになった。もちろん、明示しなければ、制御回路１により、Ｒａｎｄｏｍ方式やＬＲＵ方式などを使って自動的に決定されることになる。これは、プログラムを書くプログラマが、より細かくキャッシュを操作できるようにとの配慮からである。このような機構を持ったキャッシュの制御を、ロッカブル（ｌｏｃｋａｂｌｅ）なキャッシュ制御と呼ぶ。

【００４８】例えば、インストラクションメモリ２上にデータキャッシュ９をロックする命令を書いておく。そして、データキャッシュ８およびデータキャッシュ９上にないデータをレジスタファイル４にロードしようとすると、キャッシュミスが起き、データメモリ７からその

データをデータキャッシュ８あ るいはデータキャッシュ９にコピーする。通常は、制御回路１により、データキャッシュ８またはデータキャッシュ９のいずれにデータをコピーするかが決定されるが、この場合は、データキャッシュ９がロックされているので、データキャッシュ９へのコピーは禁止され、データキャッシュ８にコピーされる。そして、データキャッシュ８にコピーされたデータがレジスタファイル４にロードされる。

【００４９】

【発明が解決しようとする課題】例えば、図６に示すデータ（ｒｕｎ，ｌｅｖｅｌ）の例では、図１４のステップＳ６において、ａ［ｐｔｒ］およびｂ［ｐｔｒ］をレジスタファイル４にロードしなければならないが、これらのデータが、データキャッシュ８あ るいはデータキャッシュ９にコピーされていて、キャッシュミスなしにロードされるという保証はない。従って、最悪の場合、キャッシュミスを起こして、ステップＳ６における計算処理に時間がかかりすぎることが考えられる。また、ロッカブルなキャッシュ制御はプログラマに提供されてはいるが、これをどのように使えばキャッシュミスなしで高速に計算できるのかが知られていなかった。

【００５０】このように、従来、ＶＬＣを情報処理装置で計算させる場合、短い時間で処理しなければならないデータについて、キャッシュミスを起こし、データの処理に時間が長くかかり過ぎてしまう場合があ る課題があった。そのため、キャッシュミスを起こさずに確実に短い時間で処理したいという要望には答えられていなかった。

【００５１】本発明はこのような状況に鑑みてなされたものであ り、比較的短い時間内で処理しなければならないデータについては、キャッシュミスなしで高速に読み出せるようにし、その時間内で確実に処理を行うことができるようにするものであ る。

【００５２】

【課題を解決するための手段】請求項 １に記載のキャッシュメモリ制御方法は、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させることを特徴とする。

【００５３】請求項 ２に記載のキャッシュメモリ制御方法は、プログラム のうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム の所定のものをキャッシュメモリに常駐させることを特徴とする。

【００５４】請求項 ３に記載の可変長符号化方法は、第１のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第１のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常駐させることを特徴とする。

【００５５】請求項 ４に記載の可変長符号化方法は、入力データに対応する圧縮データを生成するプログラム を第２のメモリに記憶させ、第２のメモリに記憶されたプログラム のうち、プログラム が生成する圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム をキャッシュメモリに常駐させることを特徴とする。

【００５６】請求項 ５に記載の可変長復号化方法は、第１のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第１のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させることを特徴とする。

【００５７】請求項 ６に記載の可変長復号化方法は、入力データに対応する復号化データを生成するプログラムを第２のメモリに記憶させ、第２のメモリに記憶されたプログラム のうち、プログラム が生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム をキャッシュメモリに常駐させることを特徴とする。

【００５８】請求項 １に記載のキャッシュメモリ制御方法においては、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させる。従って、比較的、短時間で参照すべきデータを高速に参照することができる。

【００５９】請求項 ２に記載のキャッシュメモリ制御方法においては、プログラム のうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム の所定のものをキャッシュメモリに常駐させる。従って、比較的、短時間で実行すべきプログラム を迅速に実行させることができる。

【００６０】請求項 ３に記載の可変長符号化方法においては、第１のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第１のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常

駐させる。従って、符号長の短い圧縮データに対して高速にアクセスすることができる。

【００６１】請求項４に記載の可変長符号化方法においては、入力データに対応する圧縮データを生成するプログラムを第２のメモリに記憶させ、第２のメモリに記憶されたプログラムのうち、プログラムが生成する圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データを生成するプログラムを迅速に実行させることができる。

【００６２】請求項５に記載の可変長復号化方法においては、第１のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第１のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データに対応する復号化データに対して高速にアクセスすることができる。

【００６３】請求項６に記載の可変長復号化方法においては、入力データに対応する復号化データを生成するプログラムを第２のメモリに記憶させ、第２のメモリに記憶されたプログラムのうち、プログラムが生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データに対応する復号化データを生成するプログラムを迅速に実行させることができる。

【００６４】

【発明の実施の形態】図１は、本発明を適用した情報処理装置の一実施例の構成を示すブロック図である。図１に示した装置は、図１３に示した従来の装置の場合と基本的に同様の構成をなしている。すなわち、図１において、制御回路（ＣＯＮＴＲＯＬ）１は、装置全体を制御するようになされている。インストラクションメモリ（ＩＭＥＭ）２は、所定のインストラクション（プログラムのコード等）を記憶する。データメモリ（ＤＭＥＭ）７は、所定のデータを記憶する。レジスタファイル（ＲＥＧ）４は、制御回路１からの指令に基づいて、後述するセレクタ、およびデータキャッシュ（ＤＣＡＣＨＥ）１またはデータキャッシュ２より、データ用バス（ＤＢＵＳ）６を介して供給されたデータを記憶し、算術演算回路（ＡＬＵ）５に供給するようになされている。データ用バス６は、レジスタファイル４と算術演算回路５との間で、データの受渡しを行うようになされている。

【００６５】レジスタファイル４、算術演算回路５、データ用バス６、および制御回路１の一部分（点線で囲まれた部分）は、いわゆるＣＰＵ（Central Processing Unit）３を構成している。

【００６６】セレクタ１０は、制御回路１によって制御され、データメモリ７からのデータをデータキャッシュ８またはデータキャッシュ９のいずれかに供給したり、データキャッシュ８またはデータキャッシュ９のデータをデータメモリ７に供給するようになされている。データキャッシュ８およびデータキャッシュ９は高速にデータの読み書きが可能なメモリであり、データキャッシュ８は、セレクタ１０を介してデータメモリ７からデータが供給されたとき、制御回路１の制御により、そのデータを記憶する。また、データキャッシュ９は、セレクタ１０を介してデータメモリ７からデータが供給されたとき、制御回路１の制御により、そのデータを記憶するようになされている。

【００６７】セレクタ１１は、制御回路１の制御に基づいて、データキャッシュ８またはデータキャッシュ９のいずれかに記憶されたデータをデータ用バス６を介して、レジスタ４に供給したり、あるいは、データ用バス６を介して、レジスタファイル４からのデータをデータキャッシュ８またはデータキャッシュ９に供給するようになされている。

【００６８】インストラクションメモリ２には、処理すべきプログラムの内容（コード）が格納されている。制御回路１からの制御により、インストラクションメモリ２内のプログラムのコードが順番に読み出され、読み出されたコードを制御回路１が解釈して、適切なデータをデータメモリ７から読み出し、レジスタファイル４と算術演算回路５を使用して所定の演算を行い、データメモリ７に演算結果を戻すようになされている。

【００６９】図２は、図１の実施例において実行されるＶＬＣ（Variable Length Coding：可変長符号化）の全体のフローチャートを示している。これは、図１４に示した従来の情報処理装置におけるＶＬＣの全体のフローチャートと基本的に同様である。また、本発明においては、図１のステップＳ３１において行われる初期設定のサブルーチンが、図３に示すサブルーチンとなり、従来の例である図１５に示したものとは異なっている。

【００７０】図２に示したフローチャートのステップＳ３２乃至Ｓ３９は、図１４に示したフローチャートのステップＳ２乃至Ｓ９にそれぞれ対応しており、各ステップにおける処理内容は、図１４を参照して上述した場合と基本的に同様であるので、ここでは、ステップＳ３１の処理について説明し、その他のステップについての説明は適宜省略する。

【００７１】以下、図３のフローチャートを参照して、図１に示した情報処理装置において、初期設定が行われ

るときの動作について説明する。まず、ステップＳ４１において、図１５のステップＳ２１の場合と同様に、図７乃至図１０に示す符号を格納しておく領域ａ［０］乃至ａ［３２×４０－１］と、その符号がどれほどの長さ（ビット長）であるかを記憶しておくための領域ｂ［０］乃至ｂ［３２×４０－１］をデータメモリ７（図１）上に確保する。そして、図７乃至図１０に示す各（ｒｕｎ，ｌｅｖｅｌの絶対値）に対応する符号を、ａ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］に上位側につめて格納する。

［００７２］また、図７乃至図１０に示す各（ｒｕｎ，ｌｅｖｅｌの絶対値）に対応する符号の長さ（ビット長）をｂ［（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ］に格納する。対応する（ｒｕｎ，ｌｅｖｅｌの絶対値）が図７乃至図１０にないために、まだデータが格納されていない残りのｂ［ｊ］には２４を格納する。但し、図７乃至図１０に示す符号の最後のｓについては無視する。

［００７３］次に、ステップＳ４２において、データキャッシュ８またはデータキャッシュ９のいずれか片方をロックする。例えば、この場合、データキャッシュ９をロックする。

［００７４］次に、ステップＳ４３において、ａ［０］、ａ［１］、ａ［３２］、ａ［２］、ａ［６４］、ａ［３］、ａ［４］、ａ［３３］、ａ［５］、ａ［６］、ａ［７］、ｂ［０］、ｂ［１］、ｂ［３２］、ｂ［２］、ｂ［６４］、ｂ［３］、ｂ［４］、ｂ［３３］、ｂ［５］、ｂ［６］、およびｂ［７］（以下では、これらのデータのことを「ａ［０］、およびｂ［０］等のデータ」ということにする）をロードする。

［００７５］ここではレジスタファイル４にロードするだけで、これらのデータを使用して演算を行うことはしない。これらのデータのロードによって、最初、上述したａ［０］、およびｂ［０］等のデータは、データキャッシュ８およびデータキャッシュ９のいずれにもないのでキャッシュミスが起こり、データメモリ７からデータキャッシュ８またはデータキャッシュ９のいずれかにコピーされる。

［００７６］これは、初期値の設定であり、実際のＶＬＣを行う前段階であるから、キャッシュミスを起こして時間がいくらかかっても良い。このとき、ステップＳ４２において、データキャッシュ９がロックされているので、データメモリ７からのデータのコピーは、データキャッシュ８に対して行われる。つまり、上述したａ［０］、およびｂ［０］等のデータは全てデータキャッシュ８にコピーされる。

［００７７］ここで、上述したａ［０］、およびｂ［０］等のデータは、図７乃至図１０に示す符号のうち、７ビット（ｓを除けば６ビット）以下のビット長を有する符号である。

［００７８］次に、ステップＳ４４において、データキャッシュ８をロックし、データキャッシュ９のロックを解除する。従って、これ以降は、キャッシュミスを起こした場合、常に、データキャッシュ９にデータがコピーされる。そして、データキャッシュ８にはオーバーライトされることはないので、上述したａ［０］、およびｂ［０］等のデータはデータキャッシュ８上に常駐することになる。

［００７９］そして、このサブルーチンの処理を終了し、リターンする。

［００８０］さて、このように初期設定をすることで、例えば、図２のステップＳ３３において、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）が読み込まれると、ステップＳ３５において、ｐｔｒ＝（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ＝（１－１）×３２＋３＝３が計算され、ステップＳ３６において、ａ［３］、およびｂ［３］が読み出され、これらがそれぞれｃｏｄｅｔｍｐ、およびｃｏｄｅｌｅｎｇｔｈとされる。

［００８１］上記処理を、実際に図１に示した情報処理装置を使って計算する場合、ａ［３］、およびｂ［３］はすでにデータキャッシュ８にコピーされているので、ステップＳ３６において、キャッシュミスを起こすことなくそれらのデータａ［３］およびｂ［３］を読み出すことができる。そして、ステップＳ３８においては、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）の符号である”００１１１０”が出力される。

［００８２］この場合において、例えば５Ｍｂｉｔ（メガビット）／ｓｅｃ（秒）の一定レートで上記圧縮したデータ（符号化されたデータ）を出力しなければならないとき、ＶＬＣは、６×２００ｎｓｅｃ（ナノ秒）以内にステップＳ３３乃至Ｓ３８の処理を行わなければならない。なぜなら、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）の時は、符号長は６ビットであるからである。しかしながら、ステップＳ３６においては、キャッシュミスを起こさないので、上記時間内に計算することが可能である。

［００８３］また、例えば、図２のステップＳ３３において、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）が読み込まれると、ステップＳ３５において、ｐｔｒ＝（２－１）×３２＋１＝３３が計算され、ステップＳ３６においては、ａ［３３］、およびｂ［３３］が読み出され、これらがそれぞれｃｏｄｅｔｍｐ、およびｃｏｄｅｌｅｎｇｔｈとされる。

［００８４］上記処理を、実際に図１に示した情報処理装置を使って計算する場合、ａ［３３］、およびｂ［３３］はデータキャッシュ８にすでにコピーされているので、キャッシュミスすることなく読み出すことができる。そして、ステップＳ８においては、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）の符号である”０００１１０１”が出力される。

【００８５】この場合において、例えば５Ｍｂｉｔ／ｓｅｃの一定レートで上記圧縮したデータ（符号化されたデータ）を出力しなければならないとき、ＶＬＣは、７×２００ｎｓｅｃ以内にステップＳ３３乃至Ｓ３８の処理を行わなければならない。なぜなら、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）のときは、図７から分かるように、符号長が７ビットであるからである。しかしながら、ステップＳ３６において、キャッシュミスを起こさないので、上記時間内に計算することが可能である。ちなみに、従来の場合には、キャッシュミスを起こす可能性もあり、上記時間内に確実に計算できるとは言い切れなかった。

【００８６】また、例えば、図２のステップＳ３３において、（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）が読み込まれると、ステップＳ３５において、ｐｔｒ＝（１８－１）×３２＋１＝５４５が計算され、ステップＳ３６においては、ａ［５４５］、およびｂ［５４５］が読み出され、これらがそれぞれｃｏｄｅｔｍｐ、およびｃｏｄｅｌｅｎｇｔｈとされる。

【００８７】しかしながら、上記処理を、実際に図１の装置を使って計算するとき、ａ［５４５］、およびｂ［５４５］はデータキャッシュ８にはコピーされていない。そして、データキャッシュ９にもコピーされているとは限らない。従って、最悪の場合、キャッシュミスを起こして、データメモリ７からデータキャッシュ９へ、ａ［５４５］、およびｂ［５４５］をコピーしなければならない。そのため、この間、計算が一時中断され、コピーを終了した時点で計算が再開される。

【００８８】ステップＳ３８においては、（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）の符号である”０００００００００００１００００”が出力される。この場合、上述したようにキャッシュミスを起こすので、ステップＳ３６における処理に時間がかかり、ステップＳ３３乃至Ｓ３８の一連の処理には時間がかかることになる。しかしながら、（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）のときは、符号長は１７ビットであるから、１７×２００（＝３４００）ｎｓｅｃ以内に、ステップＳ３３乃至Ｓ３８の処理を行えば良い。従って、キャッシュミスを起こしても、上記時間内に計算することが可能である。

【００８９】このように、一定レートで圧縮したデータを出力するとき、符号長の比較的短いものに対しては比較的短い時間で計算しなければならないが、上述したように、符号長の短いものは予めデータキャッシュ８にロードし、ロックしているので、キャッシュミスを起こすことなく読み出すことができる。従って、ステップＳ３３乃至Ｓ３８の処理（計算）を制限時間内に実行することができる。

【００９０】一方、符号長の比較的長いものに対しては比較的長い時間をかけてステップＳ３３乃至Ｓ３９の処理（計算）を行っても良いので、キャッシュミスを起こしながら計算しても間に合う。このように、本発明を適用することにより、符号長が短い場合、および符号長が長い場合において、それぞれの場合で処理しなければならない時間に適した時間内で計算することができる。

【００９１】また、上述の例では、データメモリ７上に、ルックアップテーブルとしてのデータ（ａ［ｉ］、およびｂ［ｉ］（ここで、ｉ＝（ｌｅｖｅｌの絶対値－１）×３２＋ｒｕｎ））を持ち、所定の計算を行うようにしたが、以下のようにして計算することも可能である。即ち、（ｒｕｎ，ｌｅｖｅｌ）＝（３，１）のときはサブルーチン（ｓｕｂ３－１）を読み出すようにし、（ｒｕｎ，ｌｅｖｅｌ）＝（１，－２）のときはサブルーチン（ｓｕｂ１－２）を読み出すようにし、（ｒｕｎ，ｌｅｖｅｌ）＝（１，１８）のときはサブルーチン（ｓｕｂ１－１８）を読み出すようにする。

【００９２】ここで、サブルーチンｓｕｂ３－１は”００１１１ｓ”を出力させる処理を実行するサブルーチンであり、サブルーチンｓｕｂ１－２は”０００１１０ｓ”を出力させる処理を実行するサブルーチンであり、サブルーチンｓｕｂ１－１８は”０００００００００００１００００ｓ”を出力させる処理を実行するサブルーチンである。ｓは、ｌｅｖｅｌの符号（正、負）である。

【００９３】このように、図７乃至図１０に示す各（ｒｕｎ，ｌｅｖｅｌの絶対値）に対して、対応する符号を出力するサブルーチンｓｕｂ［ｒｕｎ］－［ｌｅｖｅｌの絶対値］をあらかじめ用意しておき、読み込んだ（ｒｕｎ，ｌｅｖｅｌ）に対して、サブルーチンｓｕｂ［ｒｕｎ］－［ｌｅｖｅｌの絶対値］を読み出すことにより、ＶＬＣを行うようにすることもできる。

【００９４】本発明は、このようなサブルーチンを使った場合にも適用することができる。図４は、本発明を適用した情報処理装置の他の実施例の構成を示すブロック図である。図４においては、図１において、インストラクション（この場合サブルーチン）を記憶しておくための、インストラクションキャッシュ２２およびインストラクションキャッシュ２３を新たに設けるようにしている。従って、図４に示した情報処理装置においては、データメモリ７内のデータを高速に読み出すことができるとともに、インストラクションメモリ（ＩＭＥＭ）２内のプログラム（サブルーチン）を、制御回路１に制御されたセレクタ２４を介して高速に読み出すことができるようになされている。

【００９５】また、制御回路１に制御されたセレクタ２１を介して、インストラクションキャッシュ２２またはインストラクションキャッシュ２３内に記憶されたプログラム（サブルーチン）が制御回路１に供給されるようになされている。制御回路１は、このサブルーチンに基づいて、所定の処理を実行することになる。

【００９６】上述のサブルーチンの内、例えば、７ビッ

ト以下の符号を出力させるためのサブルーチンをあらかじめインストラクションキャッシュ２２またはインストラクションキャッシュ２３にコピーさせ、ロックしておく。具体的には、サブルーチン（ｓｕｂ）０－１、ｓｕｂ１－１、ｓｕｂ０－２、ｓｕｂ２－１、ｓｕｂ０－３、ｓｕｂ３－１、ｓｕｂ４－１、ｓｕｂ１－２、ｓｕｂ５－１、ｓｕｂ６－１、およびｓｕｂ７－１の１１個のサブルーチンをインストラクションキャッシュ２２または２３にコピーし、ロックする。

【００９７】これにより、これらのサブルーチンはインストラクションキャッシュ２２またはインストラクションキャッシュ２３上に常駐しているので、キャッシュミスなしで読み出すことが可能である。つまり、高速に読み出され、制御回路１により、そのサブルーチンに従った処理が実行される。上記以外のサブルーチンは読み出されるときキャッシュミスを起こす可能性があるが、符号長が８ビット以上と長いので、キャッシュミスを起こしながら処理（計算）させても間に合う。

【００９８】以上のように、本発明は、ＶＬＣの計算において、出力する符号長は可変であり、また、出力する圧縮されたデータのレートはＭＰＥＧ２（Moving Picture Experts.Group phase 2）では一定（例えば５Ｍｂｉｔ（メガビット）／ｓｅｃ（秒））であることに着目したものである。すなわち、符号長の短い符号をＶＬＣの計算を行って出力する場合、この計算を高速に実行する必要があり、符号長の長い符号をＶＬＣの計算を行って出力する場合、この計算は比較的ゆっくり実行してもかまわない。従って、上記実施例においては、符号長に応じた適切な時間内で符号化を行うようにしている。

【００９９】また、上述の例では、ＶＬＣについて述べたが、ＶＬＤ（Variable Length Decoding：バリアブルレングスデコーディング）についても適用することが可能である。ＶＬＤとは、圧縮されたデータ（符号化されたデータ）を元に復元する操作である。具体的には、図７乃至図１０に示す符号から、（ｒｕｎ，ｌｅｖｅｌ）を生成する操作を行うものである。

【０１００】ＶＬＤは、例えば、以下のようにして計算される。即ち、入力される圧縮されたデータ（符号化されたデータ）が、”００１１１ｓ”であれば、サブルーチン（ｓｕｂｓｕｂ３－１）を読み出すようにし、”０００１１０ｓ”であれば、サブルーチン（ｓｕｂｓｕｂ１－２）を読み出すようにし、”００００００００００１０００００ｓ”であれば、サブルーチン（ｓｕｂｓｕｂ１－１８）を読み出すようにする。

【０１０１】ここで、サブルーチンｓｕｂｓｕｂ３－１は、ｒｕｎ＝３とし、ｌｅｖｅｌ＝１（この場合、ｓ＝０）、またはｌｅｖｅｌ＝－１（この場合、ｓ＝１）とするサブルーチンであり、サブルーチンｓｕｂｓｕｂ１－２は、ｒｕｎ＝１とし、ｌｅｖｅｌ＝２（この場合、ｓ＝０）、またはｌｅｖｅｌ＝－２（この場合、ｓ＝１）とするサブルーチンであり、サブルーチンｓｕｂｓｕｂ１－１８は、ｒｕｎ＝１とし、ｌｅｖｅｌ＝１８（この場合、ｓ＝０）、またはｌｅｖｅｌ＝－１８（この場合、ｓ＝１）とするサブルーチンである。

【０１０２】このように、図７乃至図１０に示す各符号に対して、対応する（ｒｕｎ，ｌｅｖｅｌ）を出力するサブルーチンｓｕｂｓｕｂ［ｒｕｎ］－［ｌｅｖｅｌの絶対値］をあらかじめ用意しておき、読み込んだ符号に対して、サブルーチンｓｕｂｓｕｂ［ｒｕｎ］－［ｌｅｖｅｌの絶対値］を読み出すことで、ＶＬＤを行うことができる。

【０１０３】本発明は、このようなＶＬＤにも適用可能である。上述したサブルーチンの内、例えば入力される符号が７ビット以下のものであるサブルーチンをあらかじめインストラクションキャッシュ２２または２３のいずれかにコピーさせ、ロックしておく。具体的には、ｓｕｂｓｕｂ０－１、ｓｕｂｓｕｂ１－１、ｓｕｂｓｕｂ０－２、ｓｕｂｓｕｂ２－１、ｓｕｂｓｕｂ０－３、ｓｕｂｓｕｂ３－１、ｓｕｂｓｕｂ４－１、ｓｕｂｓｕｂ１－２、ｓｕｂｓｕｂ５－１、ｓｕｂｓｕｂ６－１、およびｓｕｂｓｕｂ７－１の１１個のサブルーチンをインストラクションキャッシュ２２または２３のいずれかにコピーし、ロックする。

【０１０４】例えば５Ｍｂｉｔ／ｓｅｃの一定レートで圧縮したデータ（符号化されたデータ）が入力されてくる場合、符号長が８ビット以上の比較的長いものは、８×２００（＝１６００）ｎｓｅｃ以上かけて上記サブルーチンを読み出して処理を行えば良いので、キャッシュミスを起こしても間に合う。もし、高速に処理したとしても、一定レートで入力されてくるので、次の符号が入力されず単に待つだけである。また、符号長が７ビット以下の比較的短いものは、速く上記サブルーチンを読み出して処理を行わなければならないが、それに対応するサブルーチンはインストラクションキャッシュ２２または２３のいずれかに常駐しているので、キャッシュミスなしで読み出すことができ、高速に処理することが可能である。

【０１０５】また、ＶＬＤを行う場合において、上述したように、サブルーチンを用いるのではなく、ＶＬＣの場合と基本的に同様にして、例えば７ビット以下の符号長を有する符号に対応するデータ（ｒｕｎ，ｌｅｖｅｌ）を、データキャッシュ８またはデータキャッシュ９のいずれかに記憶させ、ロックする。そして、これに基づいて、ＶＬＤを行わせるようにすることも可能である。

【０１０６】なお、上記実施例においては、本発明をＭＰＥＧ２方式による符号化または復号化に適用する場合について説明したが、本発明は、これに限定されるものではない。

【０１０７】また、上記実施例は、２つのデータキャッ

シュおよび２つのインストラクションキャッシュを備えた構成としたが、上記構成に限定されるものではない。

【０１０８】また、上記実施例においては、データキャッシュに常駐させるデータを符号長が７ビット以下の符号としたが、これに限定されるものではない。同様に、インストラクションキャッシュに常駐させるデータを符号長が７ビット以下の符号に対応するサブルーチンとしたが、これに限定されるものではない。

【０１０９】また、上記実施例において、符号化または復号化の対象とするデータは、画像データに限らず、音声データやその他のデータとすることができる。

【０１１０】

【発明の効果】請求項 １に記載のキャッシュメモリ制御方法によれば、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させるようにしたので、比較的、短時間で参照すべきデータを高速に参照することができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを参照することが可能となる。これにより、データに応じた適切な時間で処理を行うことができる。

【０１１１】請求項 ２に記載のキャッシュメモリ制御方法によれば、プログラム のうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム の所定のものをキャッシュメモリに常駐させるようにしたので、比較的、短時間で実行すべきプログラム を迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラム を実行させることが可能となる。これにより、データに応じた適切な時間で処理を行うことができる。

【０１１２】請求項 ３に記載の可変長符号化方法によれば、第１のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第１のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対して高速にアクセスすることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを読み出すことが可能となる。これにより、符号長に応じた適切な時間で符号化処理を実行することができる。

【０１１３】請求項 ４に記載の可変長符号化方法によれば、入力データに対応する圧縮データを生成するプログラム を第２のメモリに記憶させ、第２のメモリに記憶されたプログラム のうち．プログラム が生成する圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム をキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データを生成するプログラム を迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラム を実行させることが可能となる。これにより、符号長に応じた適切な時間で符号化処理を実行することができる。

【０１１４】請求項 ５に記載の可変長復号化方法によれば、第１のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第１のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対応する復号化データに対して高速にアクセスすることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを読み出すことが可能となる。これにより、符号長に応じた適切な時間で復号化処理を実行することができる。

【０１１５】請求項 ６に記載の可変長復号化方法によれば、入力データに対応する復号化データを生成するプログラム を第２のメモリに記憶させ、第２のメモリに記憶されたプログラム のうち、プログラム が生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラム をキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対応する復号化データを生成するプログラム を迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラム を実行させることが可能となる。これにより、符号長に応じた適切な時間で復号化処理を実行することができる。

【図面の簡単な説明】

【図１】本発明を適用した情報処理装置の一実施例の構成を示すブロック図である。

【図２】図１に示した実施例の動作を説明するためのフローチャートである。

【図３】図２のフローチャートのステップＳ３１に対応するサブルーチンの処理を説明するためのフローチャートである。

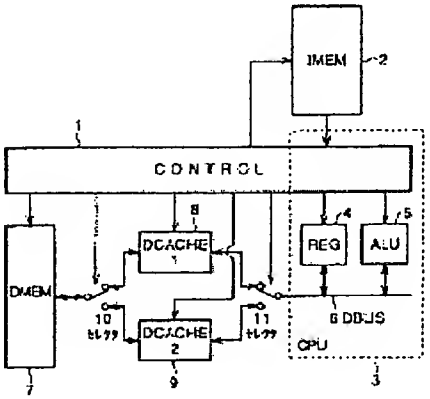
【図４】本発明を適用した情報処理装置の他の実施例の構成を示すブロック図である。

【図５】ジグザグスキャンデータ列とラインスキャンデータ列を示す図である。

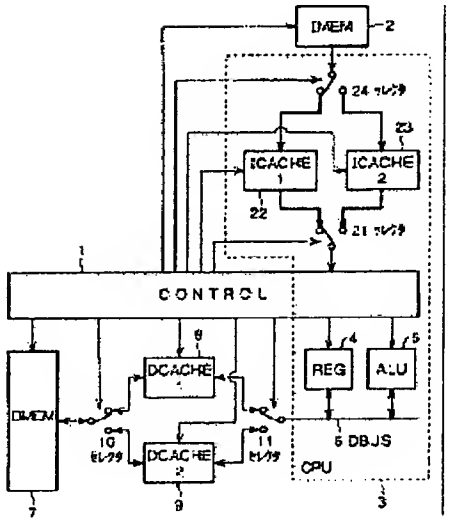
【図６】図５に対応するデータ（ｒｕｎ，ｌｅｖｅｌ）

を示す図である。
【図７】（ｒｕｎ，ｌｅｖｅｌ）対（符号）を示す図である。
【図８】（ｒｕｎ，ｌｅｖｅｌ）対（符号）を示す図である。
【図９】（ｒｕｎ，ｌｅｖｅｌ）対（符号）を示す図である。
【図１０】（ｒｕｎ，ｌｅｖｅｌ）対（符号）を示す図である。
【図１１】符号化されたデータを示す図である。
【図１２】（ｒｕｎ）対（符号）および（ｌｅｖｅｌ）対（符号）を示す図である。
【図１３】従来の情報処理装置の一例の構成を示すブロック図である。
【図１４】図１３に示した装置の動作を説明するためのフローチャートである。
【図１５】図１４のステップＳ１に対応するサブルーチンの処理を説明するためのフローチャートである。
【符号の説明】
１　制御回路
２　インストラクションメモリ
３　ＣＰＵ
４　レジスタファイル
５　ＡＬＵ
６　データ用バス
７　データメモリ
８，９　データキャッシュ
１０，１１　セレクタ
２１，２４　セレクタ
２２，２３　インストラクションキャッシュ

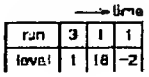
【図１】



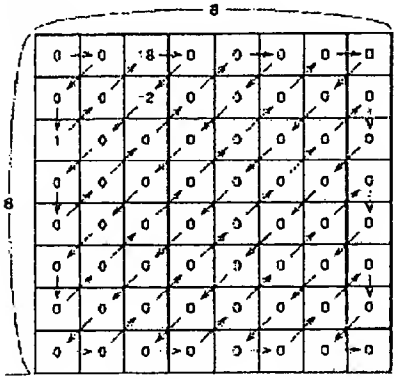
【図４】



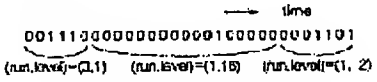
【図６】



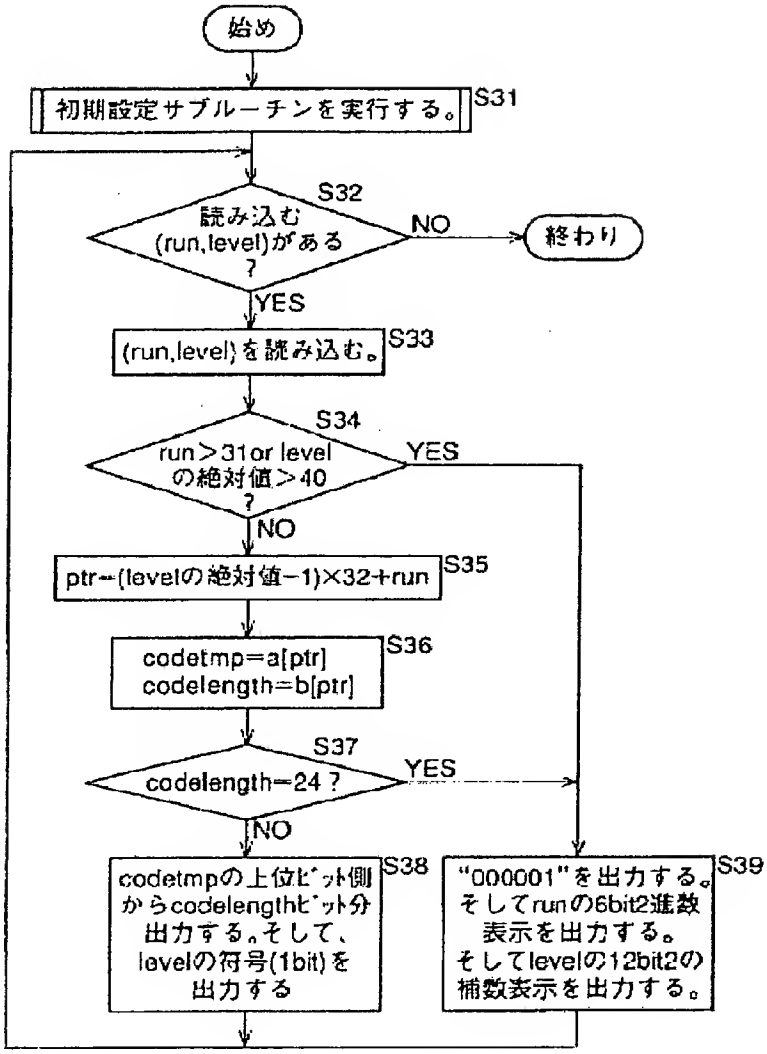
【図５】



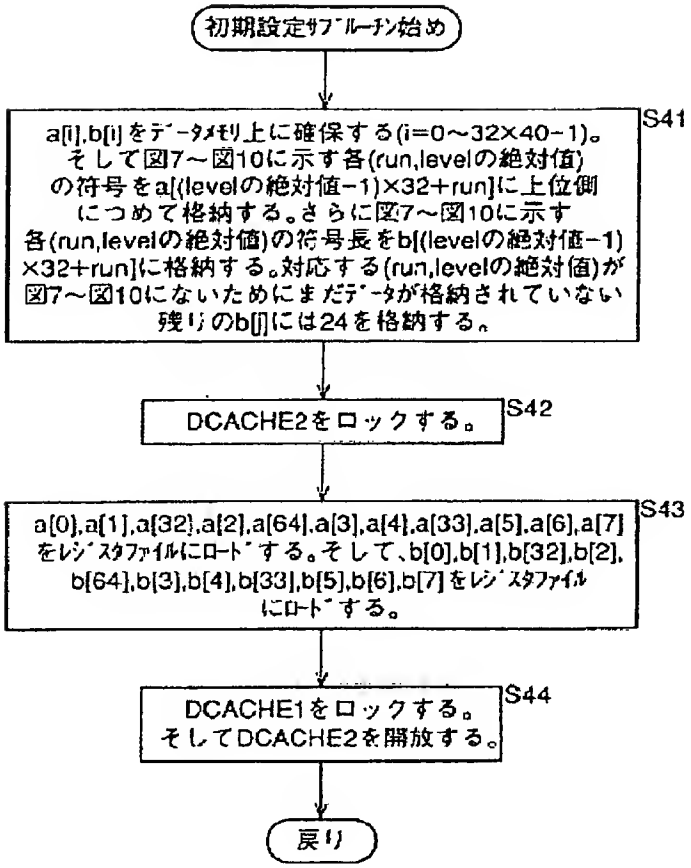
【図１１】



【図2】


始め
初期設定サブルーチンを実行する。
S31
S32
読み込む(run,level)がある？
NO
終わり
YES
(run,level)を読み込む。
S33
S34
run＞31or levelの絶対値＞40？
YES
NO
ptr=(levelの絶対値−1)×32+run
S35
codetmp=a[ptr]
codelength=b[ptr]
S36
S37
codelength=24？
YES
NO
codetmpの上位ビット側からcodelengthビット分出力する。そして、levelの符号(1bit)を出力する
S38
"000001"を出力する。そしてrunの6bit2進数表示を出力する。そしてlevelの12bit2の補数表示を出力する。
S39

【図３】



【図１２】

Table B-16 … Encoding of run and level following an ESCAPE code

fixed length code	run
0000 00	0
0000 01	1
0000 10	2
…	…
…	…
…	…
…	…
1111 11	63

fixed length code	signed level
1000 0000 0001	-2047
1000 0000 0010	-2046
…	…
1111 1111 1111	-1
0000 0000 0000	forbidden
0000 0000 0001	+1
…	…
0111 1111 1111	+2047

【図７】

Table B-14 -- DCT coefficients Table zero

Variable length code (NOTE)	run	level の絶対値	(level の絶対値 −1)×32+run
11s	0	1	0
011 s	1	1	1
0100 s	0	2	32
0101 s	2	1	2
0010 1 s	0	3	64
0011 1 s	3	1	3
0011 0 s	4	1	4
0001 10 s	1	2	33
0001 11 s	5	1	5
0001 01 s	6	1	6
0001 00 s	7	1	7
0000 110 s	0	4	
0000 100 s	2	2	
0000 111 s	8	1	
0000 101 s	9	1	
0000 01	Escape		
0010 0110 s	0	5	
0010 0001 s	0	6	
0010 0101 s	1	3	
0010 0100 s	3	2	
0010 0111 s	10	1	
0010 0011 s	11	1	
0010 0010 s	12	1	
0010 0000 s	13	1	
0000 0010 10 s	0	7	
0000 0011 00 s	1	4	
0000 0010 11 s	2	3	
0000 0011 11 s	4	2	
0000 0010 01 s	5	2	
0000 0011 10 s	14	1	
0000 0011 01 s	15	1	
0000 0010 00 s	16	1	

NOTE-The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

【図８】

Table B-14 --- DCT coefficients Table zero(continued)

Variable length code(NOTE)	run	level の絶対値
0000 0001 1101 s	10	8
0000 0001 1000 s	10	9
0000 0001 0011 s	10	10
0000 0001 0000 s	10	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	8	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1

NOTE-The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

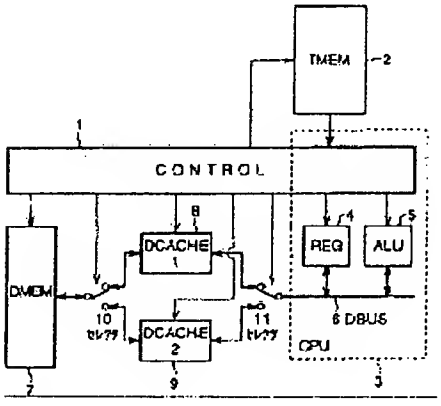
【図１０】

Table B-14 --- DCT coefficients Table zero(concluded)

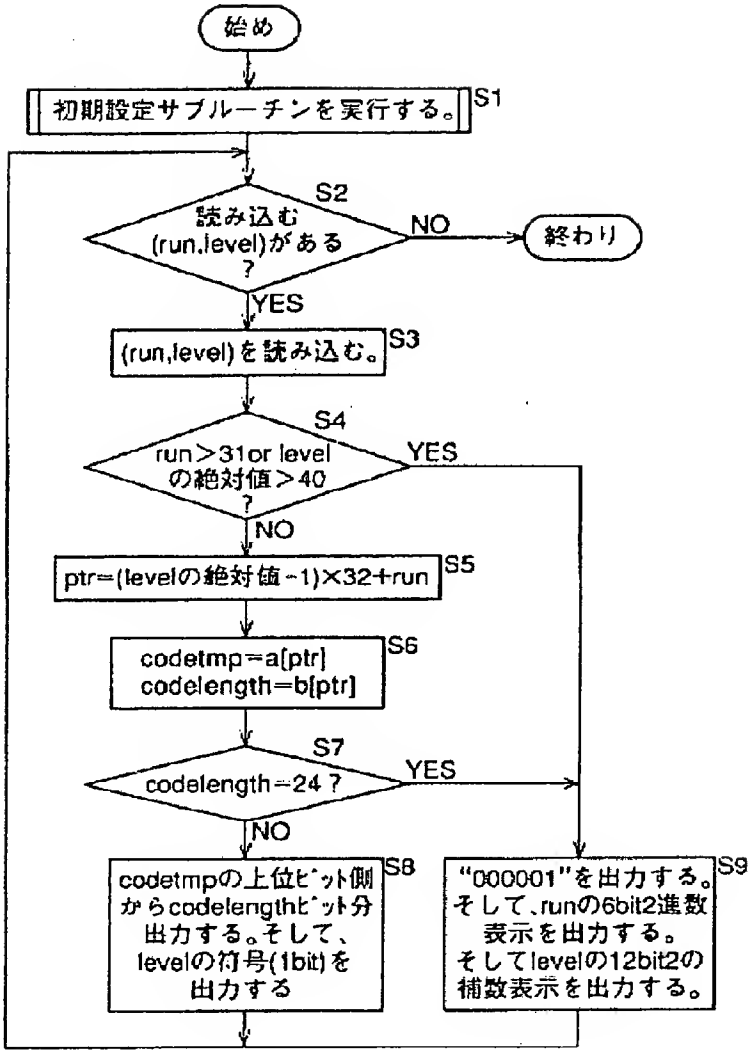
Variable length code(NOTE)	run	level の絶対値
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1

NOTE-The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

【図１３】



【図１４】


始め
初期設定サブルーチンを実行する。
S1
S2
読み込む(run,level)がある？
NO
終わり
YES
(run,level)を読み込む。
S3
S4
run＞31or levelの絶対値＞40？
YES
NO
ptr=(levelの絶対値-1)×32+run
S5
codetmp=a[ptr]
codelength=b[ptr]
S6
S7
codelength=24？
YES
NO
codetmpの上位ビット側からcodelengthビット分出力する。そして、levelの符号(1bit)を出力する
S8
"000001"を出力する。そして、runの6bit2進数表示を出力する。そしてlevelの12bit2の補数表示を出力する。
S9

【図９】

Table B-14 ··· DCT coefficients Table zero(continued)

Variable length code(NOTE)	run	levelの絶対値
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14

NOTE The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

【図１５】